PILOT-01: Bootstrap Project Structure

🎯 CONTEXT

Phase: PILOT (Week 0)
Component: Project Foundation
Estimated Time: 30 min AI execution + 20 min verification
Complexity: Medium
Risk Level: HIGH (foundation for everything)

📦 DEPENDENCIES

Must Complete First:

- **NONE** - This is the first prompt

Required Tools Installed:

bash

```bash
# Verify you have these installed:
docker --version          # Docker 20.10+
docker-compose --version  # Docker Compose 2.0+
git --version             # Git 2.30+
make --version            # GNU Make 4.0+
```

Required Environment:

- Operating System: Linux/macOS (Windows WSL2 works)
- Disk Space: 10+ GB free
- RAM: 8+ GB recommended
- Internet connection (for pulling Docker images)

🎯 OBJECTIVE

Create the **complete OptiInfra project structure** with all directories, configuration files, Docker setup, and development scripts.

Success Criteria:

✅ All directories created (services/, docs/, scripts/, .windsurf/)
✅ `docker-compose up` starts all services (PostgreSQL, ClickHouse, Qdrant, Redis)
✅ `make verify` shows all services healthy
✅ No manual fixes needed (or < 5 minutes of minor adjustments)
✅ README.md explains project structure

Failure Signs:

❌ Missing critical directories
❌ docker-compose.yml has syntax errors
❌ Services don't start or crash immediately
❌ Requires > 30 minutes of manual fixes

🏗 TECHNICAL SPECIFICATION

Project Structure to Create:

```
optiinfra/
├── README.md                   # Project overview
├── LICENSE                     # MIT License
├── .gitignore                  # Git ignore rules
├── .env.example                # Environment variables template
├── docker-compose.yml          # All services orchestration
├── Makefile                    # Development commands
│
├── services/                   # All microservices
│   ├── orchestrator/           # Go-based orchestrator
│   │   ├── cmd/
│   │   ├── internal/
│   │   ├── pkg/
│   │   ├── go.mod
│   │   ├── go.sum
│   │   ├── Dockerfile
│   │   └── README.md
│   │
│   ├── cost-agent/             # Cost optimization agent
│   │   ├── src/
│   │   ├── tests/
│   │   ├── requirements.txt
│   │   ├── Dockerfile
│   │   └── README.md
│   │
│   ├── performance-agent/      # Performance optimization agent
│   │   ├── src/
│   │   ├── tests/
│   │   ├── requirements.txt
│   │   ├── Dockerfile
│   │   └── README.md
│   │
│   ├── resource-agent/         # Resource optimization agent
│   │   ├── src/
│   │   ├── tests/
│   │   ├── requirements.txt
│   │   ├── Dockerfile
│   │   └── README.md
│   │
│   ├── application-agent/      # Application quality agent
│   │   ├── src/
│   │   ├── tests/
│   │   ├── requirements.txt
│   │   ├── Dockerfile
│   │   └── README.md
│   │
│   └── shared/                    # Shared Python utilities
│       ├── optiinfra_common/
│       ├── setup.py
│       └── README.md
│
├── portal/                        # Next.js customer portal
│   ├── src/
│   ├── public/
│   ├── package.json
│   ├── tsconfig.json
│   ├── next.config.js
│   ├── Dockerfile
│   └── README.md
│
├── docs/                          # Documentation
│   ├── ARCHITECTURE.md
│   ├── API_REFERENCE.md
│   ├── DEPLOYMENT.md
│   ├── DEVELOPMENT.md
│   └── TROUBLESHOOTING.md
│
├── scripts/                       # Utility scripts
│   ├── setup.sh                   # Initial setup
│   ├── start.sh                   # Start all services
│   ├── stop.sh                    # Stop all services
│   ├── verify.sh                  # Verify installation
│   ├── test.sh                    # Run all tests
│   └── deploy.sh                  # Deploy to production
│
├── .windsurf/                     # Windsurf prompts
│   ├── prompts/
│   │   ├── pilot/
│   │   ├── 00-foundation/
│   │   ├── 01-cost-agent/
│   │   ├── 02-performance-agent/
│   │   ├── 03-resource-agent/
│   │   ├── 04-application-agent/
│   │   └── 05-portal/
│   ├── context/
│   └── README.md
│
└── k8s/                    # Kubernetes manifests
    ├── base/
    ├── overlays/
    └── README.md
```

💻 IMPLEMENTATION REQUIREMENTS

1. docker-compose.yml (COMPLETE FILE)

yaml

```yaml
version: '3.9'

services:
  # PostgreSQL - Primary database
  postgres:
    image: postgres:15-alpine
    container_name: optiinfra-postgres
    environment:
      POSTGRES_USER: optiinfra
      POSTGRES_PASSWORD: optiinfra_dev_password
      POSTGRES_DB: optiinfra
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U optiinfra"]
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      - optiinfra-network

  # ClickHouse - Time-series metrics
  clickhouse:
    image: clickhouse/clickhouse-server:23.8-alpine
    container_name: optiinfra-clickhouse
    environment:
      CLICKHOUSE_USER: optiinfra
      CLICKHOUSE_PASSWORD: optiinfra_dev_password
      CLICKHOUSE_DB: optiinfra_metrics
    ports:
      - "8123:8123"  # HTTP interface
      - "9000:9000"  # Native interface
    volumes:
      - clickhouse_data:/var/lib/clickhouse
    healthcheck:
      test: ["CMD", "wget", "--spider", "-q", "localhost:8123/ping"]
      interval: 10s
      timeout: 5s
      retries: 5
  networks:
    - optiinfra-network

# Qdrant - Vector database for LLM memory
qdrant:
  image: qdrant/qdrant:v1.7.0
  container_name: optiinfra-qdrant
  ports:
    - "6333:6333"  # HTTP API
    - "6334:6334"  # gRPC API
  volumes:
    - qdrant_data:/qdrant/storage
  healthcheck:
    test: ["CMD", "wget", "--spider", "-q", "localhost:6333/health"]
    interval: 10s
    timeout: 5s
    retries: 5
  networks:
    - optiinfra-network

# Redis - Caching and pub/sub
redis:
  image: redis:7-alpine
  container_name: optiinfra-redis
  ports:
    - "6379:6379"
  volumes:
    - redis_data:/data
  command: redis-server --appendonly yes
  healthcheck:
    test: ["CMD", "redis-cli", "ping"]
    interval: 10s
    timeout: 5s
    retries: 5
  networks:
    - optiinfra-network

# Orchestrator (Go) - Will be added in PILOT-02
# orchestrator:
#   build:
#     context: ./services/orchestrator
```

```yaml
  #     dockerfile: Dockerfile
  #   container_name: optiinfra-orchestrator
  #   ports:
  #     - "8080:8080"
  #   depends_on:
  #     postgres:
  #       condition: service_healthy
  #     redis:
  #       condition: service_healthy
  #   networks:
  #     - optiinfra-network

  # Cost Agent (Python/FastAPI) - Will be added in PILOT-03
  # cost-agent:
  #   build:
  #     context: ./services/cost-agent
  #     dockerfile: Dockerfile
  #   container_name: optiinfra-cost-agent
  #   ports:
  #     - "8001:8000"
  #   depends_on:
  #     postgres:
  #       condition: service_healthy
  #     clickhouse:
  #       condition: service_healthy
  #     redis:
  #       condition: service_healthy
  #   networks:
  #     - optiinfra-network

volumes:
  postgres_data:
  clickhouse_data:
  qdrant_data:
  redis_data:

networks:
  optiinfra-network:
    driver: bridge
```

2. Makefile (COMPLETE FILE)

makefile

```
.PHONY: help setup dev up down restart logs verify test lint clean

# Default target
help:
	@echo "OptiInfra Development Commands"
	@echo "================================"
	@echo "make setup    - Initial setup (run once)"
	@echo "make dev      - Start all services in development mode"
	@echo "make up       - Start all services (detached)"
	@echo "make down     - Stop all services"
	@echo "make restart  - Restart all services"
	@echo "make logs     - View logs (all services)"
	@echo "make verify   - Verify all services are healthy"
	@echo "make test     - Run all tests"
	@echo "make lint     - Run linters on all code"
	@echo "make clean    - Clean up containers and volumes"

# Initial setup
setup:
	@echo "Setting up OptiInfra development environment..."
	@chmod +x scripts/*.sh
	@./scripts/setup.sh

# Start services in development mode (foreground)
dev:
	@echo "Starting OptiInfra services..."
	docker-compose up

# Start services (detached)
up:
	@echo "Starting OptiInfra services (detached)..."
	docker-compose up -d
	@sleep 5
	@make verify

# Stop services
down:
	@echo "Stopping OptiInfra services..."
	docker-compose down

# Restart services
```

```
restart:
  @make down
  @make up

# View logs
logs:
  docker-compose logs -f

# Verify all services are healthy
verify:
  @./scripts/verify.sh

# Run all tests
test:
  @./scripts/test.sh

# Run linters
lint:
  @echo "Running linters..."
  @cd services/orchestrator && go fmt ./... && go vet ./...
  @cd services/cost-agent && black src/ tests/ && flake8 src/ tests/
  @cd services/performance-agent && black src/ tests/ && flake8 src/ tests/
  @cd services/resource-agent && black src/ tests/ && flake8 src/ tests/
  @cd services/application-agent && black src/ tests/ && flake8 src/ tests/

# Clean up
clean:
  @echo "Cleaning up..."
  docker-compose down -v
  @find . -type d -name "__pycache__" -exec rm -rf {} +
  @find . -type f -name "*.pyc" -delete
  @echo "Cleanup complete"
```

3. .env.example (COMPLETE FILE)

bash

```
# OptiInfra Environment Variables
# Copy this file to .env and update with your values

# Database
DATABASE_URL=postgresql://optiinfra:optiinfra_dev_password@localhost:5432/optiinfra
POSTGRES_USER=optiinfra
POSTGRES_PASSWORD=optiinfra_dev_password
POSTGRES_DB=optiinfra

# ClickHouse
CLICKHOUSE_HOST=localhost
CLICKHOUSE_PORT=8123
CLICKHOUSE_USER=optiinfra
CLICKHOUSE_PASSWORD=optiinfra_dev_password
CLICKHOUSE_DB=optiinfra_metrics

# Qdrant
QDRANT_HOST=localhost
QDRANT_PORT=6333

# Redis
REDIS_URL=redis://localhost:6379

# Orchestrator
ORCHESTRATOR_HOST=localhost
ORCHESTRATOR_PORT=8080

# Agents
COST_AGENT_PORT=8001
PERFORMANCE_AGENT_PORT=8002
RESOURCE_AGENT_PORT=8003
APPLICATION_AGENT_PORT=8004

# LLM Configuration
OPENAI_API_KEY=sk-your-key-here
ANTHROPIC_API_KEY=sk-ant-your-key-here
LLM_PROVIDER=openai  # openai or anthropic

# Cloud Provider Credentials (for production)
AWS_ACCESS_KEY_ID=your-key-here
AWS_SECRET_ACCESS_KEY=your-secret-here
```

```
AWS_REGION=us-east-1

GCP_PROJECT_ID=your-project-id
GCP_CREDENTIALS_PATH=/path/to/credentials.json

AZURE_SUBSCRIPTION_ID=your-subscription-id
AZURE_TENANT_ID=your-tenant-id
AZURE_CLIENT_ID=your-client-id
AZURE_CLIENT_SECRET=your-secret

# Development
DEBUG=true
LOG_LEVEL=debug
ENVIRONMENT=development
```

4. scripts/setup.sh (COMPLETE FILE)

bash

```bash
#!/bin/bash
set -e

echo "🚀 OptiInfra Setup Starting..."

# Check required tools
echo "📋 Checking required tools..."

if ! command -v docker &> /dev/null; then
    echo "❌ Docker not found. Please install Docker first."
    exit 1
fi

if ! command -v docker-compose &> /dev/null; then
    echo "❌ Docker Compose not found. Please install Docker Compose first."
    exit 1
fi

if ! command -v git &> /dev/null; then
    echo "❌ Git not found. Please install Git first."
    exit 1
fi

echo "✅ All required tools found"

# Create .env from .env.example if it doesn't exist
if [ ! -f .env ]; then
    echo "📝 Creating .env file from .env.example..."
    cp .env.example .env
    echo "✅ .env file created. Please update with your values."
else
    echo "✅ .env file already exists"
fi

# Pull Docker images
echo "📥 Pulling Docker images..."
docker-compose pull

# Create network
echo "🌐 Creating Docker network..."
docker network create optiinfra-network 2>/dev/null || echo "Network already exists"
```

```
echo ""
echo "✅ Setup complete!"
echo ""
echo "Next steps:"
echo "1. Update .env with your credentials"
echo "2. Run: make dev (or make up for detached mode)"
echo "3. Run: make verify (to check all services)"
echo ""
```

5. scripts/verify.sh (COMPLETE FILE)

bash

```bash
#!/bin/bash

echo "🔍 Verifying OptiInfra Services..."
echo ""

# Colors
GREEN='\033[0;32m'
RED='\033[0;31m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

# Check PostgreSQL
echo -n "PostgreSQL... "
if docker exec optiinfra-postgres pg_isready -U optiinfra &>/dev/null; then
    echo -e "${GREEN}✅ HEALTHY${NC}"
else
    echo -e "${RED}❌ UNHEALTHY${NC}"
fi

# Check ClickHouse
echo -n "ClickHouse... "
if curl -s http://localhost:8123/ping &>/dev/null; then
    echo -e "${GREEN}✅ HEALTHY${NC}"
else
    echo -e "${RED}❌ UNHEALTHY${NC}"
fi

# Check Qdrant
echo -n "Qdrant...     "
if curl -s http://localhost:6333/health &>/dev/null; then
    echo -e "${GREEN}✅ HEALTHY${NC}"
else
    echo -e "${RED}❌ UNHEALTHY${NC}"
fi

# Check Redis
echo -n "Redis...      "
if docker exec optiinfra-redis redis-cli ping | grep -q PONG; then
    echo -e "${GREEN}✅ HEALTHY${NC}"
else
    echo -e "${RED}❌ UNHEALTHY${NC}"
```

```
fi

echo ""
echo "🎉 Infrastructure verification complete!"
```

6. README.md (COMPLETE FILE)

markdown

# OptiInfra

Multi-Agent AI Platform for Complete LLM Infrastructure Optimization

Cut costs 50% • Improve performance 3x • Ensure quality

## 🚀 Quick Start

### Prerequisites
- Docker 20.10+
- Docker Compose 2.0+
- Git 2.30+
- Make 4.0+

### Setup
```bash
# Clone repository
git clone https://github.com/yourorg/optiinfra.git
cd optiinfra

# Initial setup
make setup

# Update .env with your credentials
cp .env.example .env
# Edit .env file

# Start services
make dev
```

### Verify Installation
```bash
make verify
```

Expected output:

```
PostgreSQL... ✅ HEALTHY ClickHouse... ✅ HEALTHY Qdrant... ✅ HEALTHY Redis... ✅ HEALTHY
```

## 🏗️ Architecture

OptiInfra uses a **multi-agent architecture** with 4 specialized agents:

1. **Cost Agent** - Optimize cloud spending (spot instances, right-sizing, RIs)
2. **Performance Agent** - Improve latency and throughput (KV cache, quantization)
3. **Resource Agent** - Maximize GPU/CPU utilization
4. **Application Agent** - Monitor quality and prevent regressions

All coordinated by a **Go-based orchestrator** with intelligent routing and conflict resolution.

## 📁 Project Structure

```
optiinfra/ ├── services/ # Microservices (orchestrator, agents) ├── portal/ # Customer dashboard (Next.js) ├── docs/ # Documentation ├── scripts/ # Utility scripts ├── .windsurf/ # AI-assisted development prompts └── k8s/ # Kubernetes deployment manifests
```

## 🛠️ Development

### Start services
```bash
make dev       # Foreground mode
make up        # Detached mode
```

### View logs
```bash
make logs
```

### Run tests
```bash
make test
```

### Stop services
```bash
make down
```

## 📊 Services

Service	Port	Purpose
PostgreSQL	5432	Primary database
ClickHouse	8123/9000	Time-series metrics
Qdrant	6333	Vector database (LLM memory)
Redis	6379	Caching and pub/sub
Orchestrator	8080	Request routing
Cost Agent	8001	Cost optimization
Performance Agent	8002	Performance optimization
Resource Agent	8003	Resource optimization
Application Agent	8004	Quality monitoring

| Portal | 3000 | Customer dashboard |

## 📚 Documentation

- [Architecture](docs/ARCHITECTURE.md)
- [API Reference](docs/API_REFERENCE.md)
- [Development Guide](docs/DEVELOPMENT.md)
- [Deployment](docs/DEPLOYMENT.md)
- [Troubleshooting](docs/TROUBLESHOOTING.md)

## 🤝 Contributing

This project is currently in development. Contribution guidelines coming soon.

## 📄 License

MIT License - see LICENSE file

## 🔗 Links

- [Website](https://optiinfra.ai)
- [Documentation](https://docs.optiinfra.ai)
- [API Reference](https://api.optiinfra.ai/docs)

Built with ❤️ for the LLM infrastructure community

7. .gitignore (COMPLETE FILE)

```
# Python
__pycache__/
*.py[cod]
*$py.class
*.so
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST
*.pytest_cache
.coverage
htmlcov/
.tox/
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Go
*.exe
*.exe~
*.dll
*.so
*.dylib
*.test
```

```
*.out
vendor/
go.work

# Node
node_modules/
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.pnpm-debug.log*
.next/
out/
.vercel
.turbo

# IDEs
.vscode/
.idea/
*.swp
*.swo
*~
.DS_Store

# Environment
.env
.env.local
.env.*.local

# Docker
.docker/

# Logs
*.log
logs/

# Database
*.db
*.sqlite
*.sqlite3

# OS
```

```
.DS_Store
Thumbs.db

# Temporary
tmp/
temp/
*.tmp
```

✅ VALIDATION COMMANDS

Step 1: Create Project Structure

bash

```bash
# Run this prompt with Windsurf to create all files
# Windsurf will generate everything above
```

Step 2: Verify Files Created

bash

```bash
# Check critical files exist
ls -la README.md docker-compose.yml Makefile .env.example .gitignore

# Check directory structure
ls -la services/ portal/ docs/ scripts/ .windsurf/ k8s/

# Make scripts executable
chmod +x scripts/*.sh
```

Step 3: Setup Environment

bash

```bash
# Run setup
make setup

# Expected output:
# ✅ All required tools found
# ✅ .env file created
# ✅ Setup complete!
```

Step 4: Start Services

bash

```bash
# Start all infrastructure services
make up

# Wait for startup (30 seconds)
sleep 30
```

Step 5: Verify Services

bash

```bash
# Verify all services healthy
make verify

# Expected output:
# PostgreSQL... ✅ HEALTHY
# ClickHouse... ✅ HEALTHY
# Qdrant...     ✅ HEALTHY
# Redis...      ✅ HEALTHY
```

Step 6: Test Database Connections

bash

```
# PostgreSQL
docker exec optiinfra-postgres psql -U optiinfra -d optiinfra -c "SELECT version();"

# ClickHouse
curl http://localhost:8123/ping
# Expected: Ok.

# Qdrant
curl http://localhost:6333/health
# Expected: {"title":"qdrant - vector search engine","version":"..."}

# Redis
docker exec optiinfra-redis redis-cli ping
# Expected: PONG
```

🎯 SUCCESS CRITERIA CHECKLIST

After running all validation commands, verify:

- [ ] All directories created (services/, portal/, docs/, scripts/, .windsurf/, k8s/)
- [ ] All files exist (README.md, docker-compose.yml, Makefile, .env.example, .gitignore)
- [ ] Scripts are executable (chmod +x worked)
- [ ] `make setup` runs without errors
- [ ] `.env` file created
- [ ] `make up` starts all services
- [ ] All 4 databases are HEALTHY (verify.sh shows ✅)
- [ ] Can connect to all databases
- [ ] No errors in logs (`make logs` shows clean startup)

Expected Time: < 50 minutes total (30 min generation + 20 min verification)

🚨 TROUBLESHOOTING

Issue 1: Docker images won't pull

bash

```bash
# Solution: Check internet connection
ping google.com

# Solution: Check Docker daemon
docker info
```

Issue 2: Port conflicts (5432, 6379, etc. already in use)

bash

```bash
# Check what's using the port
lsof -i :5432

# Kill the process or change ports in docker-compose.yml
```

Issue 3: Services start but aren't healthy

bash

```bash
# View logs to see errors
docker-compose logs postgres
docker-compose logs clickhouse
docker-compose logs qdrant
docker-compose logs redis

# Common fix: Wait longer (services need 30-60s to fully start)
sleep 60 && make verify
```

Issue 4: Permission denied on scripts

bash

```bash
# Make scripts executable
chmod +x scripts/*.sh
```

📦 DELIVERABLES

This prompt should generate:

1. **Complete directory structure** (all folders)
2. **Configuration files:**
    - docker-compose.yml
    - Makefile
    - .env.example
    - .gitignore
3. **Scripts (5 files):**
    - setup.sh
    - verify.sh
    - start.sh
    - stop.sh
    - test.sh
4. **Documentation:**
    - README.md
    - Basic docs/ folder structure
5. **Working Docker setup** (4 databases running)

🎯 NEXT STEPS

After this prompt succeeds:

1. ✅ **Verify:** All services healthy
2. ✅ **Commit:** `git add . && git commit -m "PILOT-01: Bootstrap project"`
3. ➡️ **Continue:** PILOT-02 (Orchestrator Skeleton)

📝 NOTES FOR WINDSURF

IMPORTANT INSTRUCTIONS:

1. **Generate COMPLETE files** - No placeholders, no "TODO" comments
2. **Use production-ready patterns** - Real configuration, not examples
3. **Make scripts executable** - Include proper shebangs
4. **Test Docker setup** - Ensure services start correctly
5. **Create all directories** - Even empty ones (use .gitkeep if needed)
6. **Follow naming conventions** - Use exact names from specification
7. **Include proper documentation** - Clear, helpful README

DO NOT:

- Leave placeholder content
- Skip any files
- Use incorrect ports
- Create broken Docker configs
- Forget to make scripts executable

EXECUTE ALL TASKS. CREATE COMPLETE, WORKING FILES. THIS IS THE FOUNDATION FOR 69 MORE PROMPTS.